TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Intelligente Netze und Management verteilter Systeme
Prof. Anja Feldmann, Ph.D.
Gregor Maier, Jörg Wallerich, Andi Wundsam

6. Blatt Praktikum Protokolldesign WS 07/08

In diesem Aufgabenblatt geht es um die Erweiterung des P2P-Knotens vom letzten Aufgabenblatt um Peer-Suche, Suchanfragen und um die Grundlagen zum Transferieren von Dateien. Dazu folgt nun eine genauere Spezifizierung, wie die nachzurüstenden Teile zu realisieren sind.

Verarbeitung von Protokoll-Nachrichten: Für diese Aufgabe sollen Nachrichten durch einfaches Flooden weitergeleitet werden, unabhängig davon, ob sie an einen speziellen Knoten gerichtet sind oder nicht. Wenn Nachrichten generiert werden, so werden sie an *alle* Nachbarn gesendet (broadcasting). Beim Weiterleiten von Nachrichten hingegen werden Nachrichten an alle Nachbarn *außer* demjenigen gesendet, von dem die Nachricht empfangen wurde (flooding).
Bei Broadcasting/Flooding erreichen Nachrichten einen Knoten evtl. auf mehreren Wegen, wodurch ein Knoten eine Nachricht mehrfach empfangen wird. Um ein stabiles System zu erhalten, darf auf eine Nachricht aber nicht mehrfach reagiert werden, selbst wenn sie mehrfach empfangen wird. Deshalb ist es notwendig sich zu merken, welche Nachrichten man schon bearbeitet hat. Dazu dient die Nachrichten-ID (`MESSAGE-ID`). Zusammen mit der Knoten-ID (`NODE-ID`) werden Nachrichten so eindeutig identifizierbar. Ein Knoten muß sich merken, welche Nachrichten er schon bearbeitet hat um sie in Zukunft ignorieren zu können. Eine Nachricht, die neu generiert wird, sei es durch ein Benutzer-Kommando oder als Antwort, wird immer mit einer neuen ID ausgestattet.
Da die Vergabe von eindeutigen IDs in verteilten Systemen nicht trivial ist, sollen Message-IDs hier mit Hilfe eines Zählers implemetiert werden. Dadurch steigen die Message-IDs pro Knoten streng monoton an (abzüglich Neustart). Die Nachrichten-IDs können geprüft werden, indem man pro Absender Bereiche bereits aufgetauchter Nachrichten-IDs verwaltet, z.B. $1 - 11, 13 - 14$ (wir haben alle Nachrichten dieses Knotens Mit IDs von 1 bis 14 gesehen, ausser der Nachricht mit ID 12).
Damit das P2P-Netz nicht durch zu viele gleichzeitige Nachrichten überlastet wird, wird die maximale Reichweite von Nachrichten beschränkt. Dazu dient das `TTL`-Feld (time to live) einer Nachricht. Dieses Feld gibt eine Rest-Reichweite in Nachbarsprüngen wieder, die die Nachricht noch wandern darf. Die TTL wird bei der Erstellung einer Nachricht auf den Wert 3 initialisiert. Soll eine Nachricht weitergeleitet werden, so ist vor dem Senden die TTL um eins zu erniedrigen. Ist der TTL-Wert danach größer als 0, so kann die Nachricht mit der erniedrigten TTL weitergeleitet werden. Ansonsten darf die Nachricht nicht weitergeleitet werden.

Funktionalität: Folgende Funktionalität soll durch die Erweiterung unseres Protokolls im P2P-Knoten implementiert werden:

1. *Peer-Suche*: Dazu dienen die `PING` und `PONG`-Nachrichten (siehe dazu die Protokollbeschreibung auf dem 5. Übungsblatt). Wird eine Nachbar-Erkennung initiiert, soll der Knoten eine `PING`-Nachricht generieren und broadcasten. Ein Knoten, der eine solche Nachricht empfängt, soll zuerst eine `PONG`-Nachricht generieren und über den Knoten zurückleiten, von dem er die `PING`-Nachricht empfangen hat. Anschließend ist die empfangene `PING`-Nachricht (unter Berücksichtigung der TTL!) weiterzuleiten. Beachte dabei, daß `PONG`-Nachrichten gezielt an den Knoten adressiert sind, der die Peer-Suche initiiert hat. Trotzdem soll die `PONG`-Nachricht vorläufig noch per Flooding weitergeleitet werden.

   Der Knoten, der die Peer-Suche initiiert hat, soll auf alle empfangenen `PONG`-Antworten auf seine Anfrage mit einer Ausgabe der Knoten-ID und der Entfernung des antwortenden Knotens auf dem Bildschirm reagieren.

2. *Suchanfragen*: Dazu dienen die `SEARCH` und `FOUND`-Nachrichten. Suchanfragen funktionieren fast genau so wie `PING` und `PONG`, außer das `FOUND`-Nachrichten nur dann generiert werden, wenn die gesuchte Datei in einem speziellen Downloadverzeichnis vorhanden ist. Ansonsten wird die Anfrage nur weitergeleitet. Die Angabe eines Downloadverzeichnisses soll der Knotensoftware als zusätzlicher Parameter übergeben werden.

   Ein Knoten, der eine Suche initiiert, soll sich alle Knoten merken, die auf eine Suchanfrage mit `FOUND` geantwortet haben. Die Antwort enthält den Suchstring als `KEY`, so dass eine Zuordnung von Suchanfrage und Antworten möglich ist. Diese Information wird benötigt, um eine Datei zum Download anfragen zu können.

3. *Dateitransfer*: Wenn ein Knoten eine Datei herunterladen will, so sendet er eine `GET`-Nachricht ins Netz mit einem der Knoten als Ziel, die auf die entsprechende `SEARCH`-Nachricht mit einem `FOUND` geantwortet haben. Da unser Protokoll noch nicht so mächtig ist, echte Dateitransfers zu bewältigen, ist vorläufig mit einer Fehlermeldung vom Typ `510 NOT IMPLEMENTED` zu antworten.

   Beim Hochladen einer Datei wird eine `PUT`-Nachricht an den Zielknoten gesendet. Auch hier soll dieser vorläufig mit `510 NOT IMPLEMENTED` antworten.

Die Protokollnachrichten, die zum Realisieren dieser Funktionalität benötigt werden dürfen nur auftauchen, *nachdem* der `HELLO`-Handshake abgeschlossen ist. Dafür dürfen keine `HELLO`-Nachrichten mehr auftauchen, nachdem der Handshake abgeschlossen wurde. Sollte eine Nachricht zu einem Zeitpunkt empfangen werden, zu dem sie noch nicht oder nicht mehr erlaubt ist, so stellt dies eine Protokollverletzung dar und die Verbindung muss nach dem Senden einer entsprechenden Fehlermeldung abgebrochen werden.

Benutzer-Schnittstelle: Um Peer-Suche, Suche und Dateitransfer starten zu können, soll die Knotensoftware entsprechende Kommandos von der Tastatur lesen können. Solche Kommandos ***könnten*** sein:

ping Weist die Software an, ein *Ping* ins Netz zu senden und so die Peer-Suche zu starten.

search <**filename**> Weist die Software an, eine Suchanfrage ins Netz zu senden.

get <**filename**> Weist die Software an, eine Datei aus dem Netz herunterzuladen.

put <**filename**> <**node-id**> Weist die Software an, eine Datei auf den angegebenen Knoten hochzuladen.

Aufgabe 1: (100 Punkte) **P2P System, Teil 2**

Erweitere die Knoten-Software um Peer-Suche, Informations-Suche und Datentransfer. Dazu gehören:

(a) (10 Punkte) Kommando-Eingaben

(b) (10 Punkte) Neue Nachrichten-Typen und Reaktion auf Protokollverletzungen

(c) (40 Punkte) Erkennung von bereits empfangenen Nachrichten

(d) (40 Punkte) Weiterleiten von Nachrichten durch Fluten (flooding) unter Berücksichtigung der TTL

Abzugeben sind:

- Dein Programm (sowohl Quelltext als auch ggf. compiliert im Falle von C, C++ und Java)
- Ggf. eine Beschreibung, was funktionieren sollte, dies leider aber immer noch nicht tut...

Details zur Abgabe der Aufgaben: siehe FAQ
(unterhalb `http://www.net.t-labs.tu-berlin.de/teaching/ws0708/PD_labcourse/`).
Abgabedatum: Dienstag, 4. Dezember 2007, 11:59h s.t.